



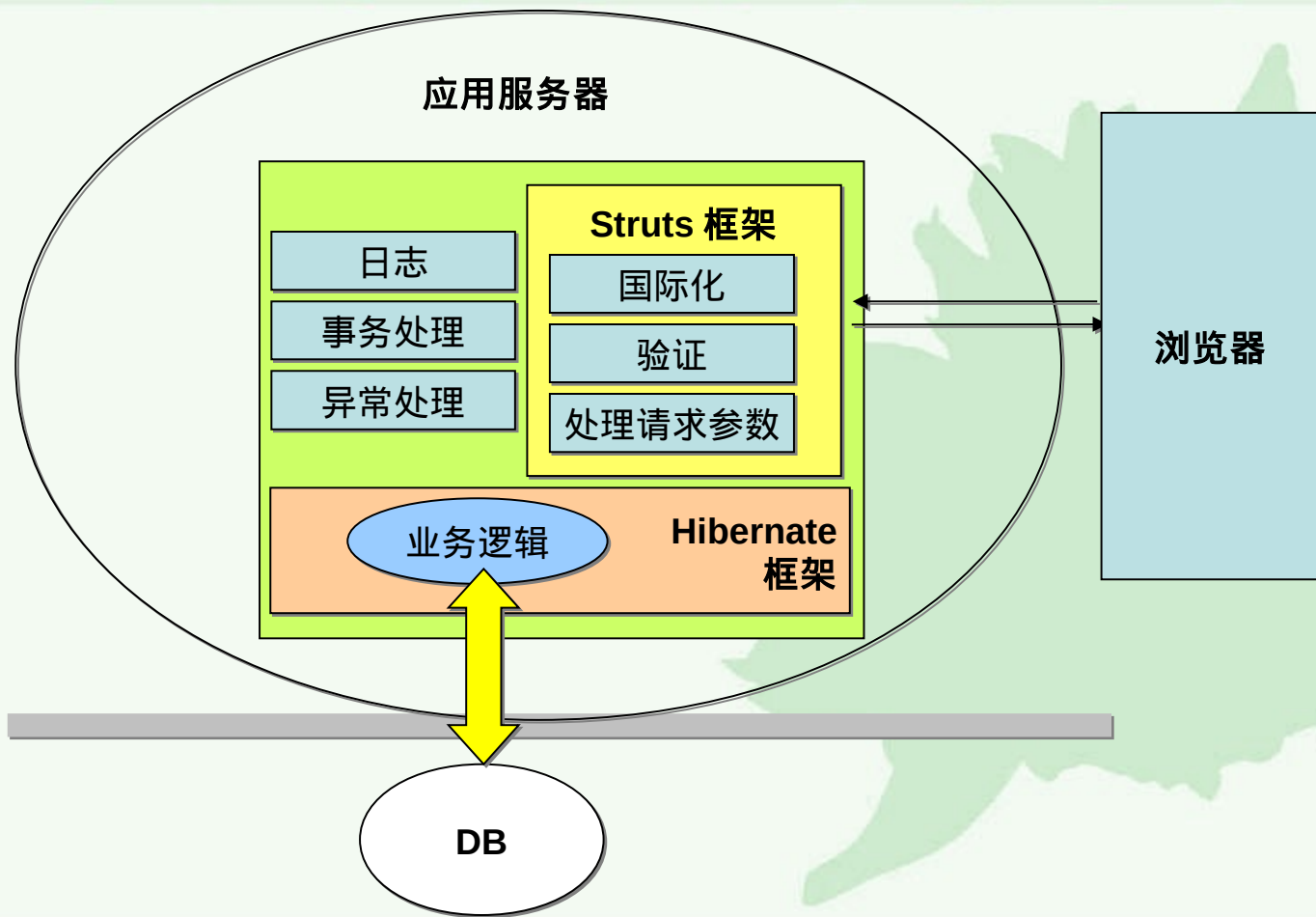
WEB

第 11 章 第三方开发框架 SSH

本章目录

- 11.1 Struts 框架
- 11.2 Hibernate 框架
- 11.3 Spring 框架
- 11.4 本章小结

示例



- **Struts** 框架主要侧重实现 MVC 中的**视图层**和**控制器层**，它提供封装的一系列类、JSP 自定义标签库以及消息资源，并将这些组件整合到统一的框架中。
- **Spring** 框架的作用则贯穿了整个大系统，它将**基于 Hibernate 的模型层**（ DAO、 PO ）和**基于 Struts 的控制器层**进行了无缝整合。
- **Hibernate** 是一个对象关系映射框架，它对 JDBC 进行了轻量级的对象封装，使得 Java 程序员可以使用对象编程的思维来操纵数据库。模型层以 DAO 设计模式为基础，所有的 DAO 类都使用 **Hibernate** 与数据库进行交互处理。

Struts 框架

Struts 框架

Struts 是 Apache 软件基金下 Jakarta 项目的一部分。Struts 在英文中是“支架、支撑”的意思，这表明了 Struts 在 Web 应用开发中的巨大作用，采用 Struts 可以更好地遵循 MVC 模式。Struts 框架的应用使开发更加规范、统一。经过长达五年的发展，Struts 已经逐渐成长为一个稳定、成熟的框架，并且成为了世界上应用最广泛的 MVC 框架之一。

Struts 框架

- **视图层** : Struts 的视图层采用 JSP 实现。 Struts 提供了基于 JSP 的一系列客户化标签库以及 ActionForm。客户化标签库可以简化创建用户界面的过程，最大限度地减少脚本的使用。
- **控制器层** : 由核心控制器层和业务逻辑控制器层两个部分组成，核心控制器主要是 ActionServlet，对于业务逻辑的操作则主要由 Action、ActionMapping、ActionForward 这几个组件协调完成。其中，Action 扮演了真正的业务逻辑的实现者，而 ActionMapping 和 ActionForward 则指定了不同业务逻辑或流程的运行方向。
- **模型层** : 与标准 MVC 项目中的模型层几乎是一样的，也是通过类似 DAO 的方式封装底层的数据库访问等业务逻辑。 Hibernate 框架可以取代原有的 JDBC 操作，采用持久化的方式进行处理。

Struts 的内置类

- **ActionServlet** : Struts 的控制器由中央控制器与业务逻辑控制器两部分所组成。ActionServlet 便是**中央控制器**，是**Struts 中最核心的类**，所有以指定后缀（如 *.do ）的 URL 请求都是先通过它进行处理。
- **ActionForm** : 一种特殊的 JavaBean 。用于**在视图层和控制器之间传递表单数据的 Java 对象**。由于 ActionForm 类是抽象类，因此我们必须在应用中创建它的子类来对应具体的 HTML 表单，**创建的类的属性必须与表单的控件名称一一对应**。

Struts 的内置类

- **Action** : 是 Struts 的**业务逻辑控制器**。当用户请求某个 Action 后，ActionServlet 就会调用请求处理器 RequestProcessor 类的 processActionPerform() 方法，而又调用 Action 实例的 execute() 方法。
- **ActionMapping** : **存储与用户请求对应的**

```
<action attribute="loginForm"
        input="/login.jsp"
        name="loginForm"
        path="/login"
        scope="request"
        type="struts.action.LoginAction">
    <forward name="success" path="/success.jsp" redirect="true"
/>
</action>
```

Struts 的内置类

- **ActionForward** : 代表一个 JSP 等 Web 资源路径的抽象表示形式。其中, **redirect 的属性值如果是 true**, 那么 **ActionForward 实例将进行响应重定向, 反之则进行请求转发。**
- 以下代码是 struts-config.xml 中当前 Action 映射信息中的 forward 元素:

```
<forward name="success" path="/success.jsp"
redirect="true" />
```

- 要获取以上的 ActionForward 实例, 可以在当前 Action 中使用 ActionMapping 的 findForward() 方法 代码如下:

```
return mapping.findForward("success");
```

Struts 标签库

- Struts 标签库是联系视图组件和 Struts 框架中其它组件的纽带，这些标签可以**访问或显示来自于控制器和模型组件的数据**。

标 签 库	说 明
html 标签	用来创建能够和 Struts 框架的 ActionForm 对应的表单以及和其它 HTML 标签相应的标签
bean 标签	用于访问 JavaBean、输出属性值、输出消息及定义请求参数等
logic 标签	管理条件产生的输出和对象集产生的循环
tiles 标签	随着 Tiles 框架包的出现，此标记已开始减少使用

Struts 标签库 (html 标签库)

- ◆ `<html:link>`
- ◆ `<html:form>`
- ◆ `<html:radio>`
- ◆ `<html:multibox>`
- ◆ `<html:submit>` 和 `<html:reset>`
- ◆ `<html:img>`
- ◆ `<html:text>`
- ◆ `<html:checkbox>`
- ◆ `<html:select>` 和 `<html:option>`
- ◆ `<html:error>`

Struts 标签库 (html 标签库)

- **<html:link>** 可以直接用 page 属性指向一个网页文件。代码如下：

```
<html:link page="index.jsp"> welcome</html:link>
```

- 此行代码解析后为：

```
<a href="index.jsp">Click demo</a>
```

- **<html:link>** 也可以用 forward 属性和 Struts 配置文件中的 **<global-forwards>** 元素中的 **<forward>** 子元素匹配。

Struts 标签库（html 标签库）

- **<html:img>** 的主要属性是 page，用于指定图象文件的路径，其它属性与基本 HTML 中的 的属性类似。代码如下：

```
<html:img page="logo.gif" height="50" width="200" />
```

- **<html:form>**：每一组 <html:form> 标签必须对应一个自定义的 ActionForm，且 <html:form> 中必须包含一个 action 属性，它是**这个标签中唯一必需的属性**。此外，此标签中不需要设置 method 属性，默认的提交方式就是 POST。代码如下：

```
<html:form action="/login" >
```

Struts 标签库 (html 标签库)

- **<html:text>** 对应于 type 属性值为“text”的 HTML<input> 标签。每一个标签都有一个 property 属性，**属性的值必须和 ActionForm 子类中相应的属性拥有同样的名子**。基本用法如下：

```
<html:text property="name" />
```

- **<html:radio>** : 如果 ActionForm 子类的某个属性**有多个可枚举的选择值**，就可以使用 <html:radio> 来采集用户的输入信息。代码如下：

```
男<html:radio property="sex" value="1" />  
女<html:radio property="sex" value="2" />
```

- 其中两个 <html:radio> 中的 **property 属性值必须都为“sex”**，value 表示选中当前选项后，提交给服务端的值

Struts 标签库 (html 标签库)

- **<html:checkbox>** 基本用法如下：

```
<html:checkbox property="student" value="true" />
```

- **<html:multibox>**

所有 **property** 属性值相同的 **<html:multibox>** 标签被映射到同一个 **property** 所指的属性中，并且这个属性是一个数组类型。而 **<html:checkbox>** 标签只对应了一个值。 **<html:multibox>** 标签的基本用法如下：

```
上网 <html:multibox property="hobbies" value=" 上网 " />  
旅游 <html:multibox property="hobbies" value=" 旅游 " />  
足球 <html:multibox property="hobbies" value=" 足球 " />
```


Struts 标签库 (html 标签库)

- **<html:select> 和 <html:option>** 基本用法如下 :

```
<html:select property="work" >
  <html:option value=" 软件工程师 " />
  <html:option value=" 软件测试工程师 " />
  <html:option value=" 其他 " />
</html:select>
```

- **<html:submit> 和 <html:reset>** : value 属性表示在按钮上显示的信息。基本用法如下 :

```
<html:submit value=" 提交 " />
<html:reset value=" 重置 " />
```

```
<html:errors property="username"/>
```

• **<html:errors>** 标签检查 Request 内的 ActionErrors 中是否存在对应的 ActionMessage 对象。代码如下 :

Struts 标签库 (bean 标签库)

- ◆ `<bean:parameter>`
- ◆ `<bean:define>`
- ◆ `<bean:write>`
- ◆ `<bean:message>`

Struts 标签库 (bean 标签库)

- **<bean:parameter>**

用于获得 HTTP 请求参数的值，并创建一个 page 范围的变量来保存所获得的 HTTP 请求参数的值。

- ◆ 该标签具有三个常用属性：

- **id** 用于保存 HTTP 请求参数值的变量名
- **name** 代表请求中的参数名
- **value** 代表如果 name 所指的参数不存在时的默认值

- 在 beantest.jsp 中用 <bean:parameter> 获取该参数值：

Struts 标签库 (bean 标签库)

- **<bean:define>**

用来将 Java 对象的属性值保存在变量中。

- ◆ **<bean:define>** 标签有五个常用属性：

- **id** 代表变量名
- **name** 代表 Java 对象名
- **property** 代表 Java 对象属性名
- **scope** 代表要获取的 name 所指的对象所在的范围
- **toScope** 代表 id 所指的变量要保存的范围

- **Scope** 和 **toScope** 属性如果不指定，默认都是 page 范围

Struts 标签库 (bean 标签库)

<bean:define>

- 以下代码在默认的 page 范围中声明了一个字符串类型的变量 user ，它的值为“ mary”：

```
<bean:define id="username" value="mary" />
```

- 以下代码在默认的 page 范围中声明了一个字符串类型的变量 username ，它的值为 session 中的 person 对象的 username 属性值：

```
<bean:define id="username" name="person"  
property="username" scope="session" />
```

Struts 标签库 (bean 标签库)

- **<bean:write>**

- 用于输出字符串变量及对象变量的属性值。
- 以下代码可以输出上文中 <bean:parameter> 和 <bean:define> 所声明的变量“username”的值

```
Username is: <bean:write name="username" />
```

- 如果要直接获取 session 中的 person 对象的

```
Username is:  
<bean:write name="person" property="username"  
scope="session" />
```

:

Struts 标签库 (bean 标签库)

- **<bean:message>**

- 将所有要显示的文字统一在消息资源文件中进行定义，最终在 JSP 中采用 <bean:message> 标签从消息资源文件中获得字符串信息并输出。标签中最常用的属性是 key，**key 对应消息资源文件中的某一行字符串信息的键名。**
- 例如在 Struts 的默认消息资源文件中有如下自定义的字符串信息：

```
index.welcome =welcome to this site!
```

- 在首页的 JSP 中如果要输出欢迎信息，可以用如下代码：

```
<bean:message key ="index.welcome" />
```


Struts 标签库（ logic 标签库 ）

在 Struts 应用中， logic 标签库主要用于根据特定的逻辑条件来判断网页内容、循环遍历集合元素以及进行请求转发和重定向等功能。以下是 logic 标签库的常用标签：

- ◆ `<logic:empty>` 与 `<logic:notEmpty>`
- ◆ `<logic:present>` 与 `<logic:notPresent>`
- ◆ `<logic:equal>` 与 `<logic:notEqual>`
- ◆ `<logic:greaterEqual>` 与 `<logic:greaterThan>`
- ◆ `<logic:lessEqual>` 与 `<logic:lessThan>`
- ◆ `<logic:match>` 与 `<logic:notMatch>`
- ◆ `<logic:iterator>`
- ◆ `<logic:forward>`
- ◆ `<logic:redirect>`

Struts 标签库 (logic 标签库)

- **<logic:empty> 与 <logic:notEmpty>**

- <logic:empty> 用于判断对象是否为 null、是否是一个空的字符串、是否是一个空的 collection 或 map。

```
<logic:empty name="myBean">  
The bean is missing  
</logic:empty>
```

- 以上代码表示当一个名为 myBean 的 bean 在所有的 scope 中都不存在时，输出 The bean is missing
- <logic:notEmpty> 的含义与 <logic:empty> 相反，使用方法类似。

Struts 标签库（ logic 标签库 ）

- **<logic:present> 与 <logic:notPresent>**

- <logic:present> 用于检查 parameter、cookie、JavaBean 对象或 JavaBean 属性是否存在且不等于 null

```
<logic:present name="person" property="username"
scope="session">
```

```
    The bean property person.username is present
```

```
</logic:present>
```

- 以上代码检查在 session 作用域内名为 person 的 bean 是否有一个 username 属性。
- <logic:notPresent> 的含义与 <logic:present> 相反，使用方法类似。

Struts 标签库（ logic 标签库 ）

- **<logic:equal> 与 <logic:notEqual>**

- **<logic:equal> 用于比较是否相等。**

```
<logic:equal name="person" property="state" value="1">  
在线  
</logic:equal>
```

- **如果上例中的 value 值需要动态获得，可采用 JSP 表达式来解决问题：**

```
<logic:equal name="charge" property="num" value="<  
%=buz.getNum()%>">  
</logic:equal>
```

Struts 标签库（ logic 标签库 ）

<logic:greaterEqual> 与 <logic:greaterThan>

- <logic:greaterEqual> 为大于等于比较符。
- 当某学生的成绩大于等于 90 时，输出“优秀”：
- <logic:greaterThan> 为大于比较符，使用方法同 <logic:greaterEqual>。

```
<logic:greaterEqual name="student" property="score"
value="90">
    优秀
</logic:greaterEqual>
```

• <logic:lessEqual> 与 <logic:lessThan>

- <logic:lessEqual> 为小于等于比较符，使用方法同 <logic:greaterEqual>。

Struts 标签库 (logic 标签库)

- **<logic:match> 与 <logic:notMatch>**

- <logic:match> 比较对象是否相等。
- 检查在 request 范围内的 name 属性值是否包含“Tom”：
- <logic:notMatch> 与 <logic:match> 含义相反，使

```
<logic:match name="name" scope="request" value="Tom">  
<bean:write name="name"/> 中有“Tom”。  
</logic:match>
```

- **<logic:redirect>**

- 该标签用于重定向，并可直接指定 URL 以及传
递参数：

```
<logic:redirect href="http://www.163.com"/>
```

Struts 标签库 (logic 标签库)

- **<logic:iterator>**

- 用于显示 collection 的值 (List、HashMap 等)
- 逐一输出列表 (userList) 中用户的姓名和年龄

```
<logic:iterate id="user" name="userList">  
  < bean:write name="user" property="name"/>  
  < bean:write name="user" property="age"/><br>  
< /logic:iterate>
```

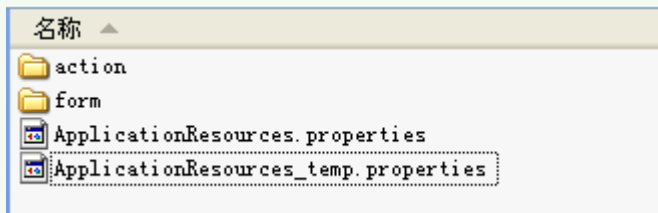
- **<logic:forward>**

- 该标签用于实现页面导向，查找 Strust 配置文件中

```
<logic:forward name="index"/>
```

Struts 的国际化处理

1. 建立临时中文消息资源文件



2. 对中文消息资源文件进行编码

输入“native2ascii”命令及其参数进行编码转换

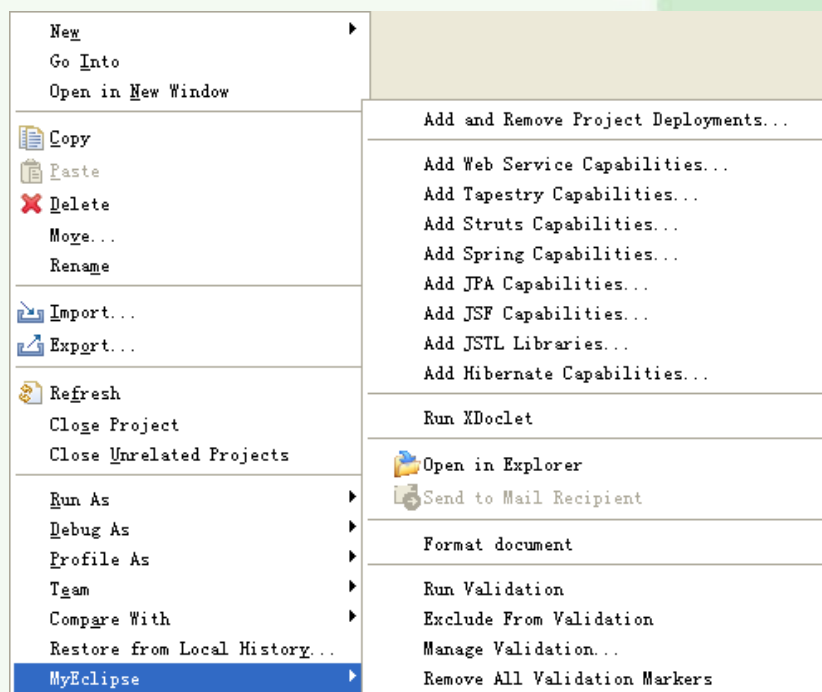
```
native2ascii -encoding gb2312  
ApplicationResources_temp.properties  
ApplicationResources_zh_CN.properties
```

3. 定义 JSP 页面的字符集

```
<%@ page language="java" pageEncoding="UTF-8"%>
```

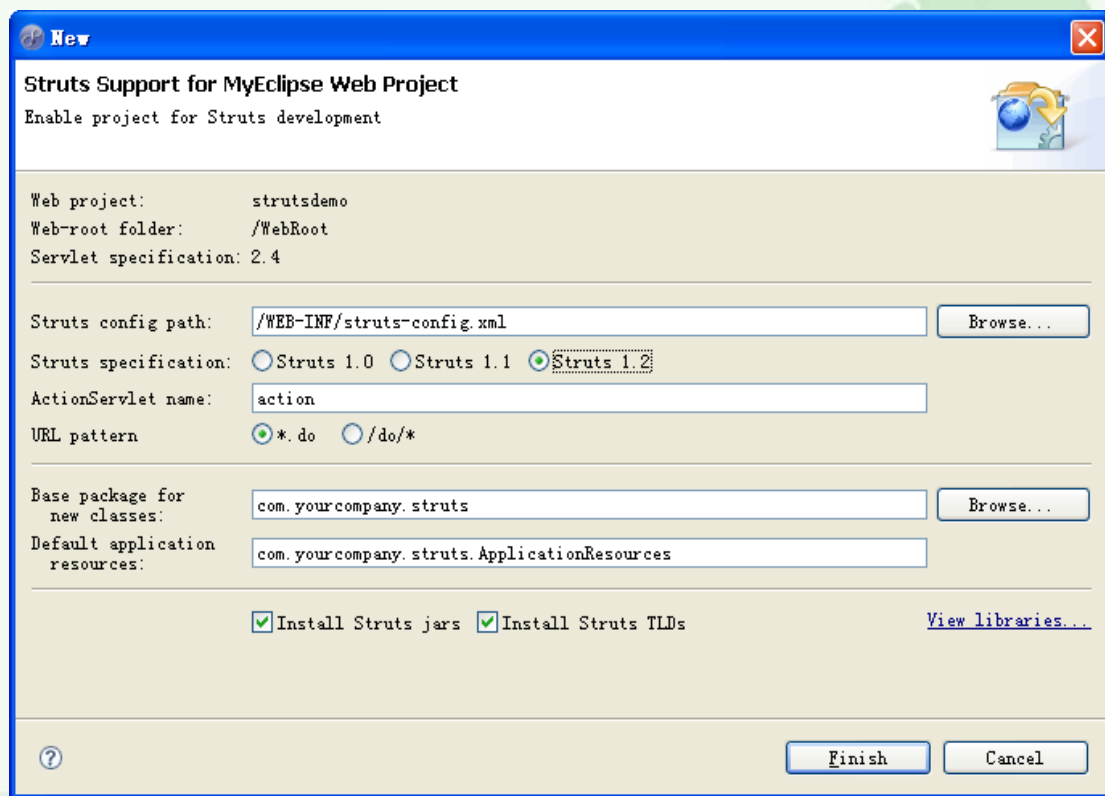

Struts 实例开发

1. 建立支持 Struts 框架的 Web 项目



Struts 实例开发

- 配置 Struts 的项目属性



The screenshot shows a 'New' dialog box titled 'Struts Support for MyEclipse Web Project'. The dialog is used to configure a web project for Struts development. It includes fields for the web project name, web-root folder, servlet specification, Struts configuration path, Struts specification, ActionServlet name, URL pattern, base package for new classes, and default application resources. There are also checkboxes for installing Struts jars and TLDs, and a link to view libraries.

New Struts Support for MyEclipse Web Project
Enable project for Struts development

Web project: strutsdemo
Web-root folder: /WebRoot
Servlet specification: 2.4

Struts config path: /WEB-INF/struts-config.xml Browse...

Struts specification: ☐ Struts 1.0 ☐ Struts 1.1 ☒ Struts 1.2

ActionServlet name: action

URL pattern: ☒ *.do ☐ /do/*

Base package for new classes: com.yourcompany.struts Browse...

Default application resources: com.yourcompany.struts.ApplicationResources

☒ Install Struts jars ☒ Install Struts TLDs [View libraries...](#)

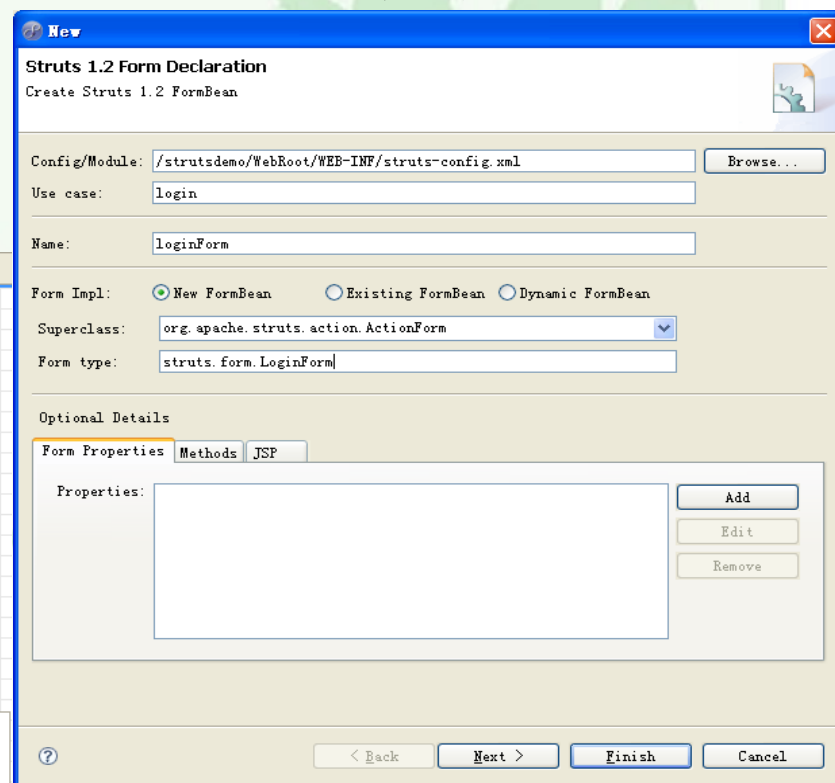
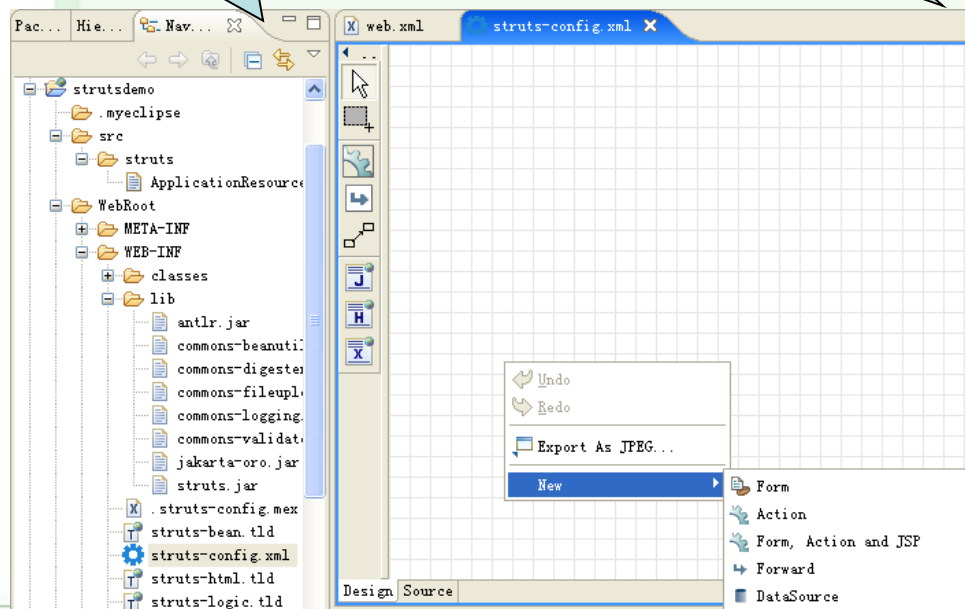
? Finish Cancel

Struts 实例开发

2. 采用向导模式创建基本的 Struts 应用

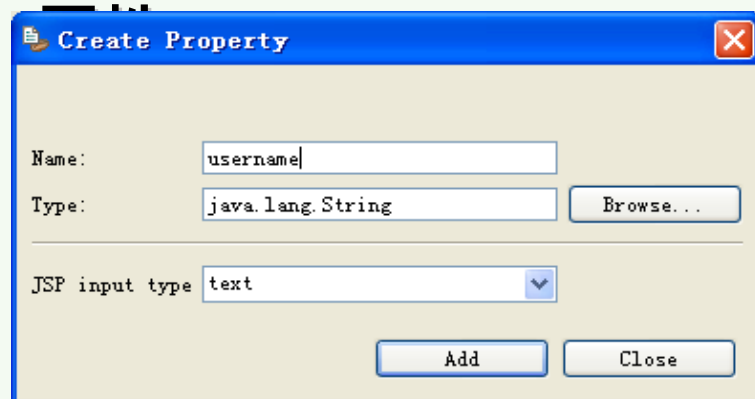
struts-config.xml 的
设计界面

新建 ActionForm
的配置向导



Struts 实例开发

- 添加名为“username”和“password”字符串类型的

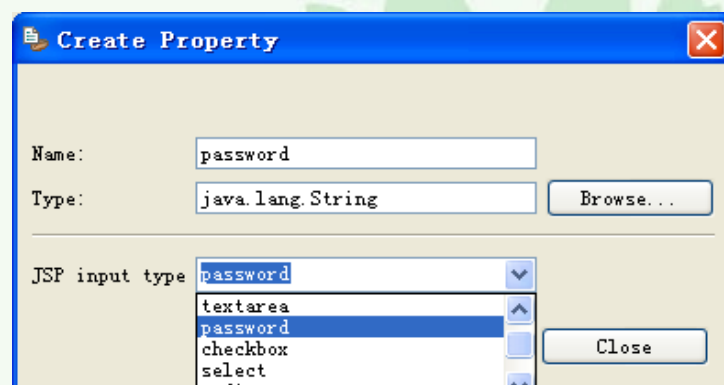


Dialog: Create Property

Name:

Type:

JSP input type:

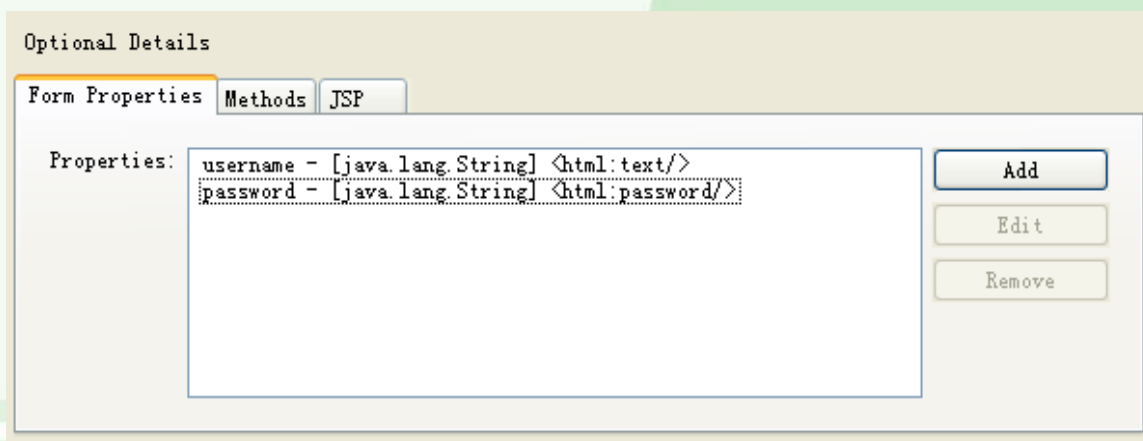


Dialog: Create Property

Name:

Type:

JSP input type:



Optional Details

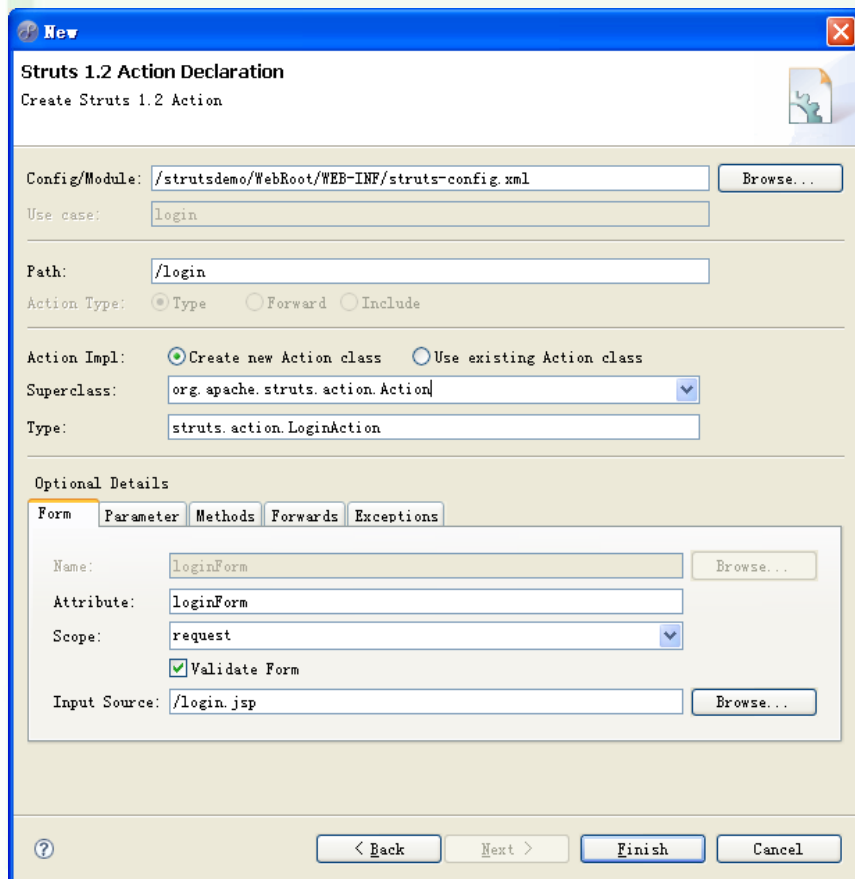
Form Properties Methods JSP

Properties:

username	-	[java.lang.String]	<html:text/>
password	-	[java.lang.String]	<html:password/>

Struts 实例开发

- 新建 Action 的配置向导



Struts 1.2 Action Declaration
Create Struts 1.2 Action

Config/Module:

Use case:

Path:

Action Type: ☒ Type ☐ Forward ☐ Include

Action Impl: ☒ Create new Action class ☐ Use existing Action class

Superclass:

Type:

Optional Details

Form **Parameter** **Methods** **Forwards** **Exceptions**

Name:

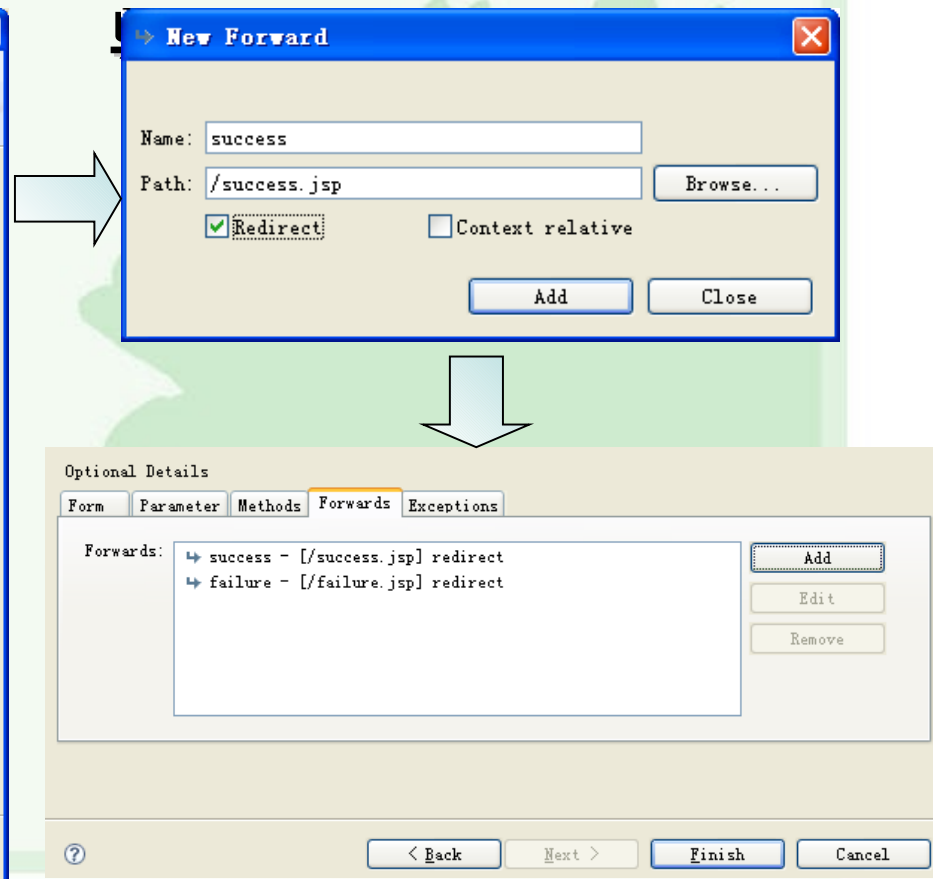
Attribute:

Scope:

☒ Validate Form

Input Source:

- 新建 ActionForward 向导



New Forward

Name:

Path:

☒ Redirect ☐ Context relative

Optional Details

Form **Parameter** **Methods** **Forwards** **Exceptions**

Forwards:

- success - [/success.jsp] redirect
- failure - [/failure.jsp] redirect

Struts 实例开发

3. 完成 LoginForm 的 validate 方法

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {
    ActionErrors errors=new ActionErrors() ;
    if(username.equals(""))
    {
        errors.add("username",new
        ActionMessage("nullerror.username")) ;
    }
    if(password.equals(""))
    {
        errors.add("password",new
        ActionMessage("nullerror.password")) ;
    }
    return errors ;
}
```

}

Struts 实例开发

4. 配置消息资源文件

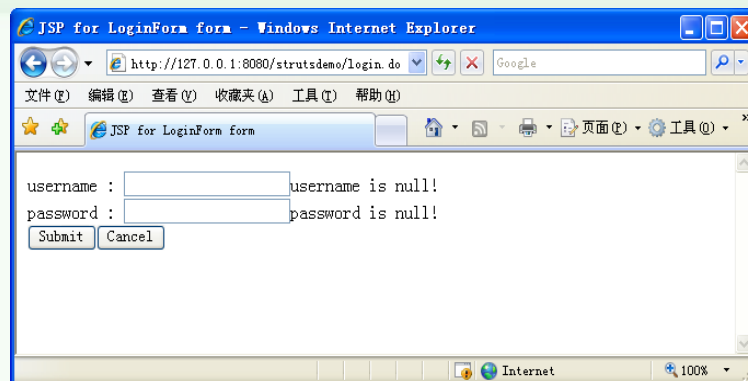
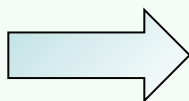
在 Struts 的消息资源文件 `ApplicationResources.properties` 中，我们添加如下代码：

```
nullerror.username=username is null!  
nullerror.password=password is null!
```

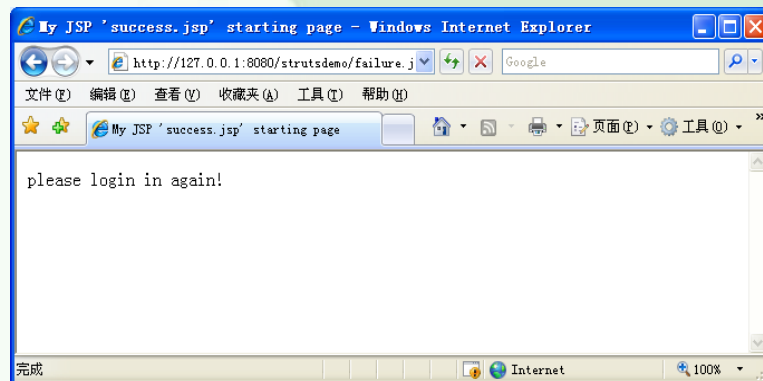
5. 完成 `LoginAction` 的 `Execute` 方法

```
public ActionForward execute(ActionMapping mapping, ActionForm form,  
                             HttpServletRequest request, HttpServletResponse  
response) {  
    LoginForm loginForm = (LoginForm) form ;  
    if (loginForm.getUsername().equals("admin")  
        &&  
        loginForm.getPassword().equals("123456")) {  
        return mapping.findForward("success") ;  
    }  
    return mapping.findForward("failure") ;  
}
```

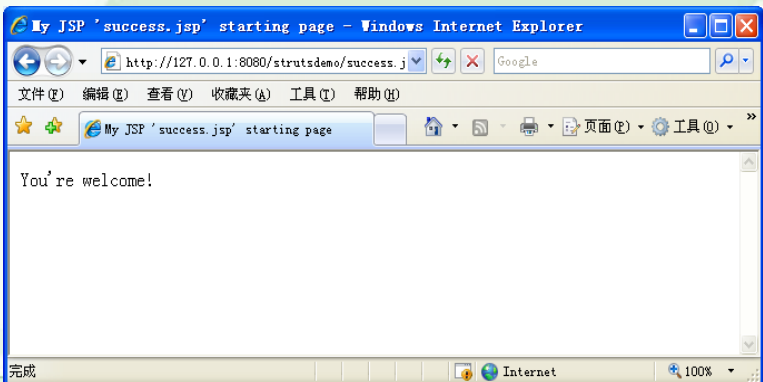
- 用户名和密码都不填



- 输入错误的用户名和密码



- 输入正确的用户名和密码

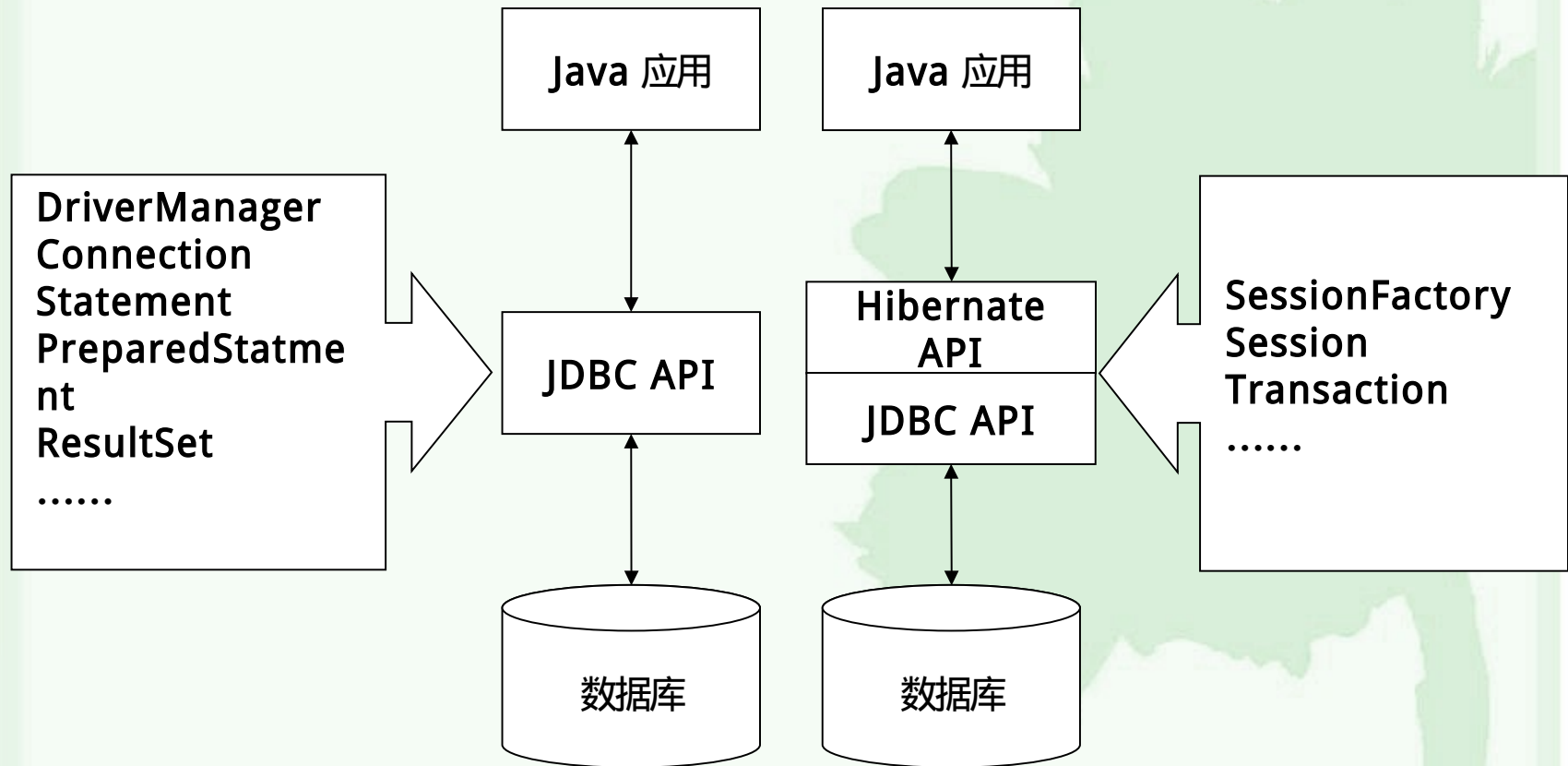


Hibernate 框架

Hibernate 概述

- **Hibernate** 是一个面向 Java 环境的对象 / 关系数据库映射工具，它能消除那些针对特定数据库厂商的 SQL 代码，并且把结果集由关系表的形式转换成对象的形式
- **Hibernate** 不仅可以管理 Java 对象（持久化对象）到数据库表的映射，还提供数据查询的方法，可以大幅度地减少在开发时人工使用 SQL 的时间。

通过 JDBC API 和 Hibernate API 不同方式访问数据库



Hibernate 的内置接口

- **Session** : 负责执行被持久化对象的 CRUD 操作 (Create、 Read、 Update 和 Delete)。此外，要注意的是 **Hibernate 的 Session 不同于 Web 应用中的 HttpSession**。
- **SessionFactory** : 充当数据存储源的代理，并负责创建 Session 对象。
- **Configuration** : 该接口负责配置并启动 Hibernate，创建 SessionFactory 对象。
- **Transaction** : 负责事务相关的操作。
- **Query 和 Criteria** : 负责执行各种数据库查询。

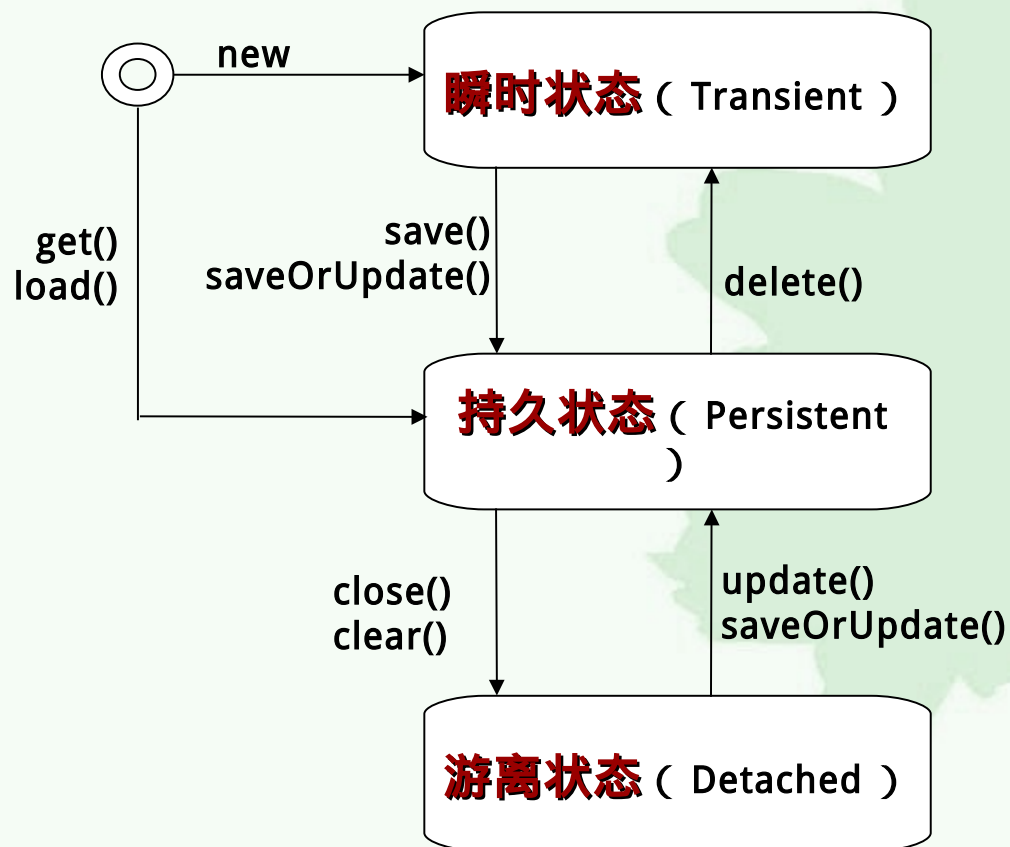
Hibernate 的持久化操作

- 持久化是指把数据保存到可永久保存的存储设备中，以面向对象的方式对数据库执行增、删、改操作

```
public class Person implements java.io.Serializable {  
    private Integer id;  
    private String name;  
        public Person() {  
    }  
    public Person(Integer id, String name, Integer age) {  
        this.id = id; this.name = name; this.age = age;  
    }  
    public Integer getId() {  
        return this.id;  
    }  
    public void setId(Integer id) {  
        this.id = id;  
    }  
    .....  
}
```

}

持久化对象的三种状态以及状态间的转换



HQL 查询方式

- HQL 是完全面向对象的查询语言，因此可以支持面向对象中的继承和多态等机制。Hibernate 中内置了 Query 类，每个 Query 实例对应一个查询对象。
- 使用 HQL 查询按如下步骤进行：
 1. 编写 HQL 语句；
 2. 以 HQL 语句作为参数，调用 Session 的 createQuery 方法创建查询对象；
 3. 如果 HQL 语句包含参数，调用 Query 的 setXxx 方法为参数赋值；
 4. 调用 Query 对象的 list 等方法返回查询结果。

HQL 查询方式

Query 包含以下常用方法

返回类型	方 法	功 能
Query	setFirstResult(int firstResult)	设置返回的结果集从第几条记录开始
Query	setMaxResults(int maxResults)	设置本次查询返回的结果数
Query	setParameter(int position,Object val)	设置 HQL 语句需要传入的参数
List	list()	返回结果集

```
String queryString = "from Person where Person.age = ?";  
Query queryObject = Session.createQuery(queryString);  
queryObject.setParameter(0,20);  
List personList=queryObject.setFirstResult(0)  
.setMaxResults(10)  
.list();
```

HQL 的常用语法

➤ from 子句：

```
from Person as p
```

➤ select 子句：

```
select p.name.firstName from Person as p
```

➤ 聚集函数：

```
select count(*) from Person  
select max(p.age) from Person as p
```

➤ where 子句：

```
from Person where name like "jack"
```


条件查询方式

- 条件查询是 Hibernate 中更具面向对象特色的数据查询方式，主要通过 **Criteria** 和 **Criterion** 两个接口以及其它相关辅助类来实现。
- 执行条件查询的步骤如下：
 1. 以 Session 对象创建 Criteria 对象。
 2. 增加 Criterion 查询条件。
 3. 执行 Criteria 的 list 方法返回结果集。

条件查询方式

- **Criteria**

- Criteria 接口代表一次查询，该查询本身并不具备任何的数据筛选功能，需要 Session 调用 `createCriteria(Class class)` 方法对某个持久化类创建条件查询的实例。

- **Criterion**

- Criterion 是 Criteria 的查询条件。Criteria 提供了 `add(Criterion criterion)` 方法来添加查询条件。Criterion 对象可以由 Restrictions 类负责产生，而 Restrictions 是专门用于产生查询条件的工具类，它的方法大部分都是静态方法

条件查询方式

- **按逻辑组合查询条件来限制结果集**

- **查询年龄值为 0 或者为空的且姓“陈”的人员信息：**

```
List PersonList= session.createCriteria(Person.class)
    .add(Restrictions.like("name", " 陈%"))
    .add(Restrictions.or(
        Restrictions.eq("age",new Integer(0)),
        Restrictions.isNull("age")))
    .list();
```

- **按指定属性列对结果集排序**

- **查询 50 个姓“陈”的人员信息，结果按年龄值倒序排**

```
List PersonList=    session.createCriteria(Person.class)
    .add(Restrictions.like("name" , " 陈%"))
    .addOrder(Order.desc("age"))
    .setMaxResults(50)
    .list();
```

条件查询方式

- **Projection**

- Projection 接口主要是让 Criteria 能够实现分组统计查询、子查询等功能。Projection 主要由 **ProjectionList** 和 **Property** 等实现类

-

- 以下代码按年龄分组统计人数：

```
List results = session.createCriteria(Person.class)
    .setProjection(Projections.projectionList()
    .add(Projections.rowCount(),"personCount")
    .add(Projections.groupProperty("age"),"age"))
    .list();
```

条件查询方式

- **DetachedCriteria**

- 离线查询 DetachedCriteria 类可以在一个 Session 范围之外创建一个查询，并且可以使用任意的 Session 来执行它。DetachedCriteria 也可以用以表示子查询。示例代码如下：

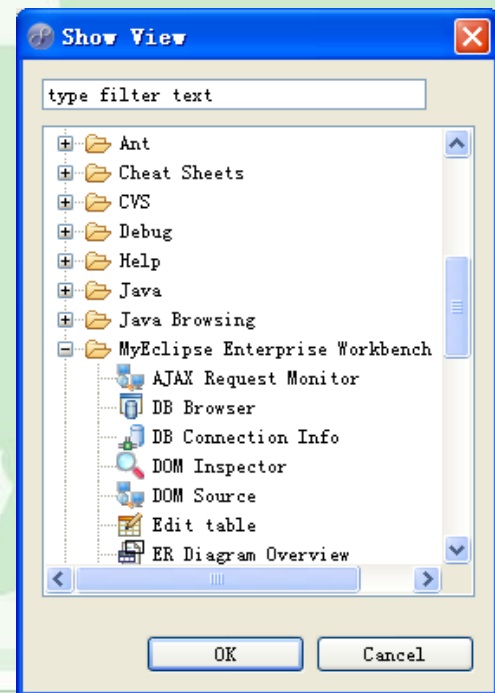
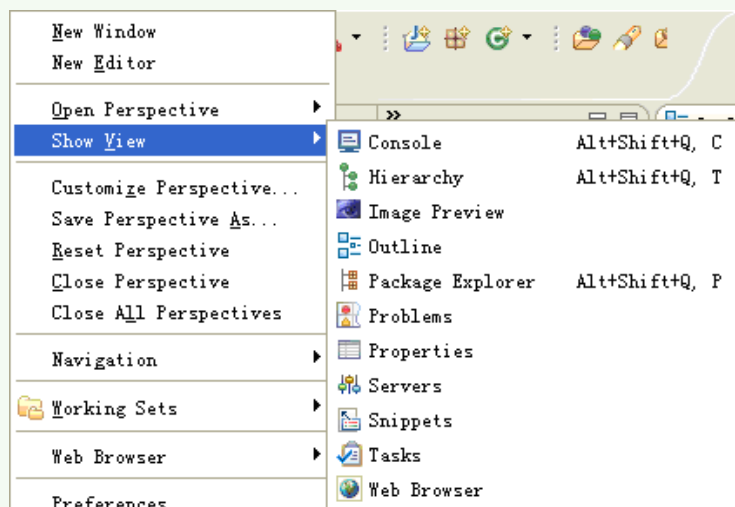
```
DetachedCriteria avgAge =  
DetachedCriteria.forClass(Person.class)  
    .setProjection(Property.forName("age").avg());  
List results = session.createCriteria(Person.class)  
    .add(Property.forName("age").gt(avgAge))  
    .list();
```

- 以上代码首先创建一个平均年龄的子查询对象，然后将其作为外层查询的比较条件进行查询，得到的结果为年龄大于平均年龄的人员信息。

Hibernate 实例开发

一、用 DB Browser 连接数据源

1. 点击菜单【Window | Show View | Other】
2. 选择【MyEclipse Enterprise Workbench | DB Brower】

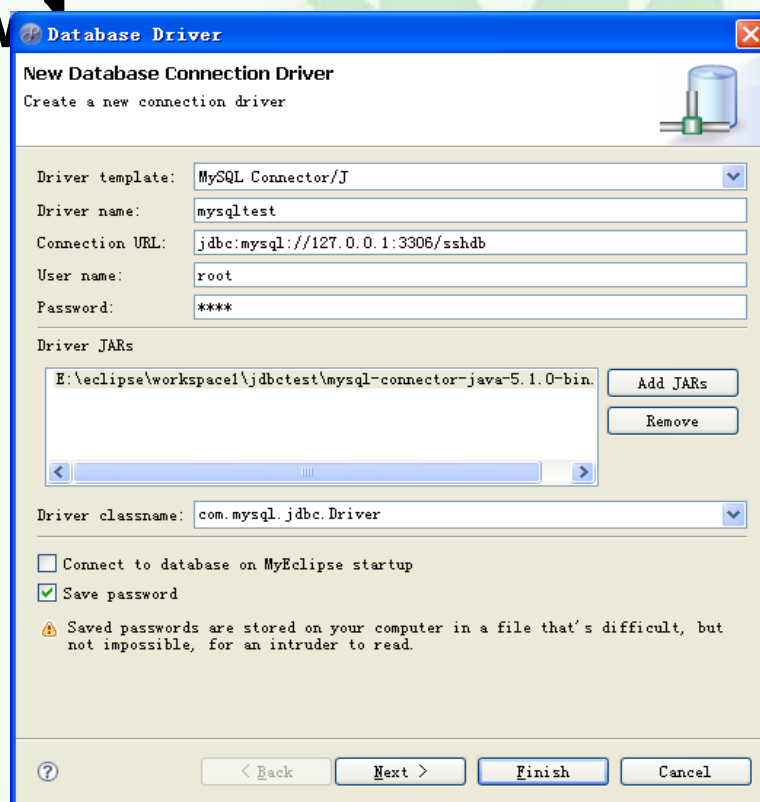
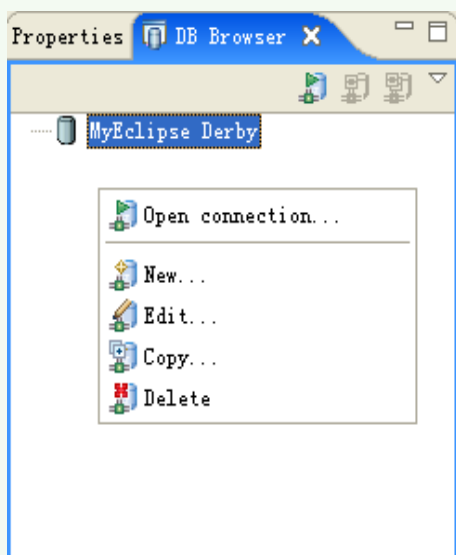


Hibernate 实例开发

一. 用 DB Browser 连接数据源

3. 鼠标右击菜单中的【New...】

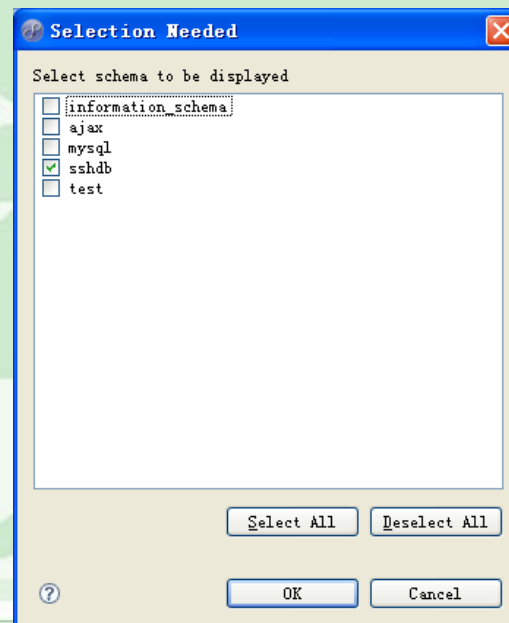
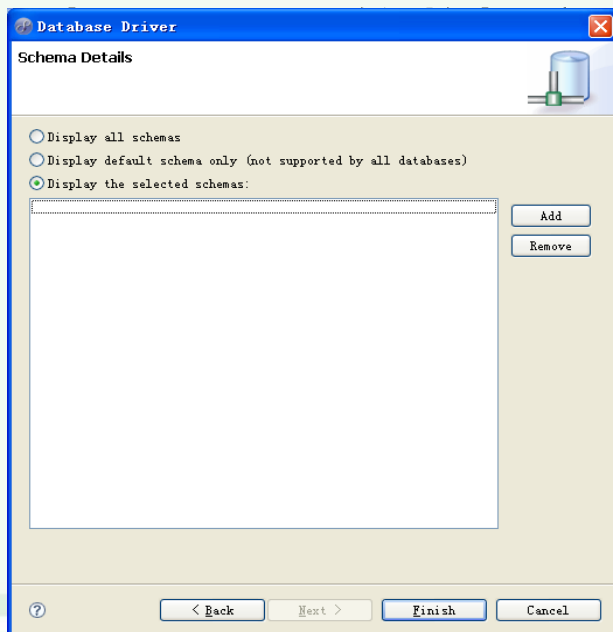
4. 完成数据库连接配置向导



Hibernate 实例开发

一. 用 DB Browser 连接数据源

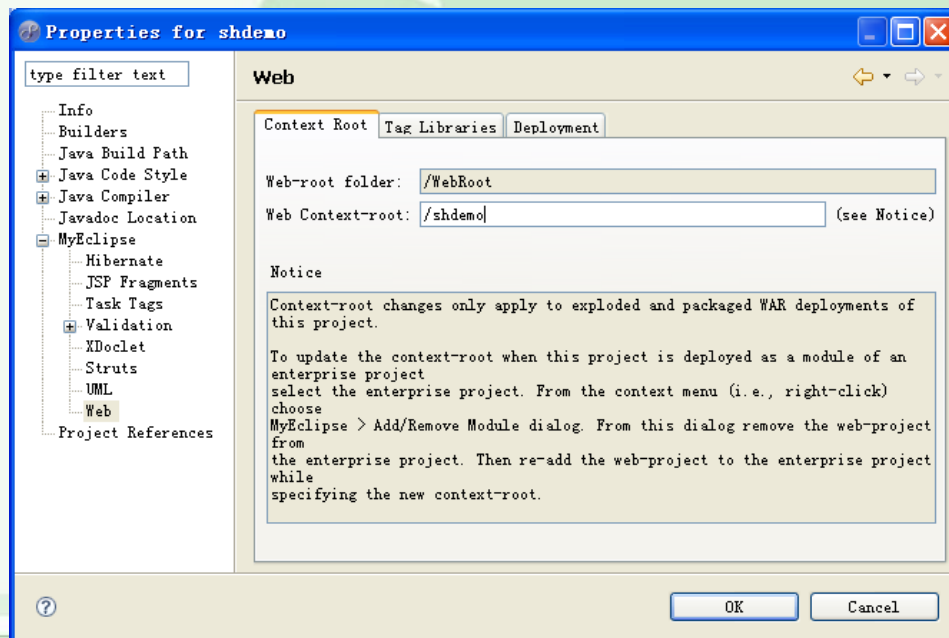
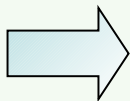
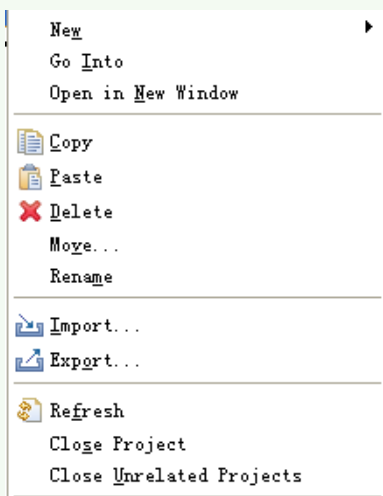
5. 点击【 Next 】，选择“ Display the selected schemas”
6. 继续点击右侧的【 Add 】，选择其中已经建立的名



Hibernate 实例开发

二. 创建支持 Struts 框架的 Web 项目

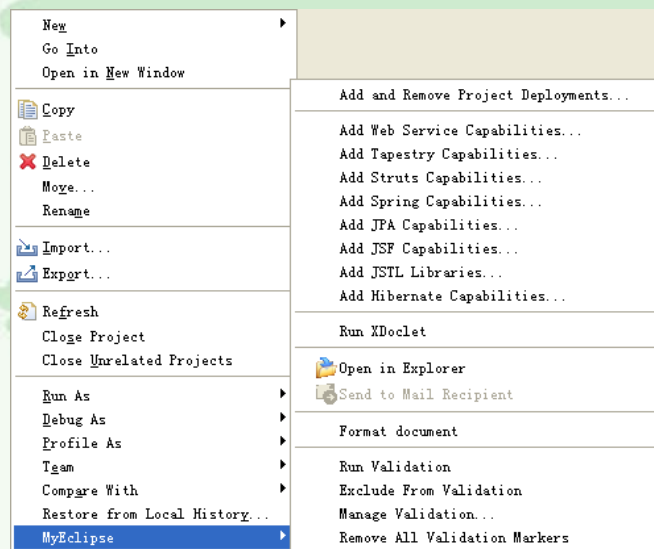
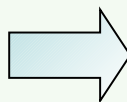
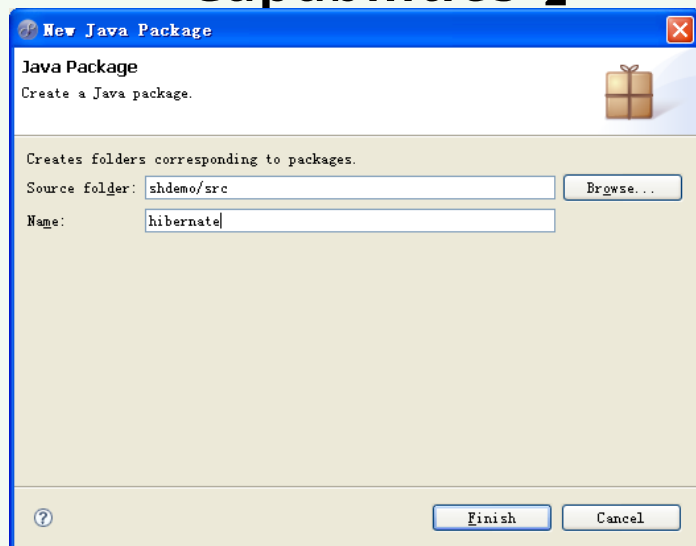
1. 复制 struts 项目，新项目起名为“shdemo”
2. 选择【Properties】，【MyEclipse | Web】，将 Web Context-root 的值修改为“/shdemo”



Hibernate 实例开发

三、为当前项目添加对 Hibernate 框架的支持

1. 在当前项目的 src 文件夹下建立名为“hibernate”的包
2. 右键菜单选择【MyEclipse | Add Hibernate Capabilities】

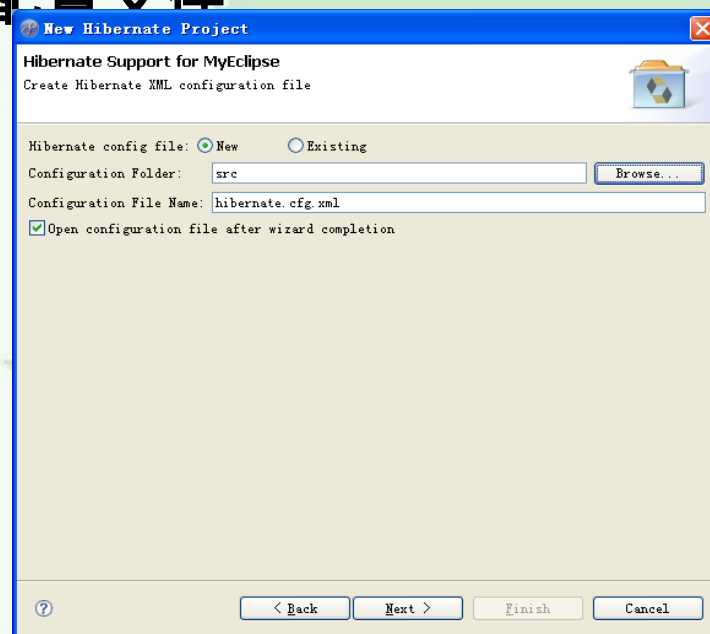
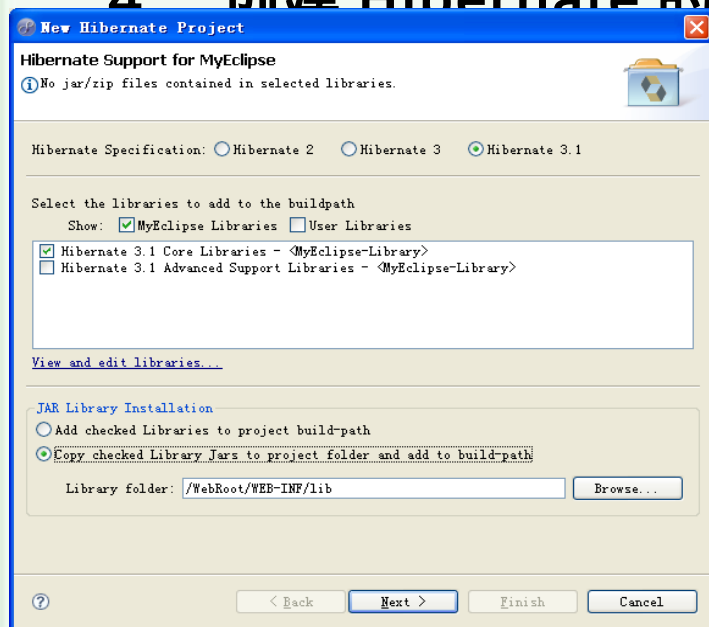


Hibernate 实例开发

三、为当前项目添加对 Hibernate 框架的支持

3. 将核心 JAR 包添加到项目的“ /WebRoot/WEB-INF/Lib” 下

4. 创建 Hibernate 的 XML 配置文件

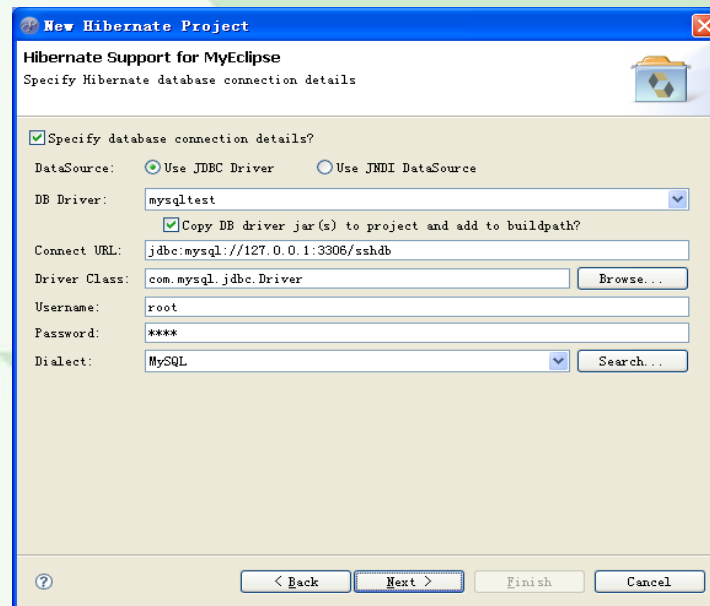
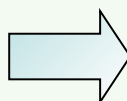
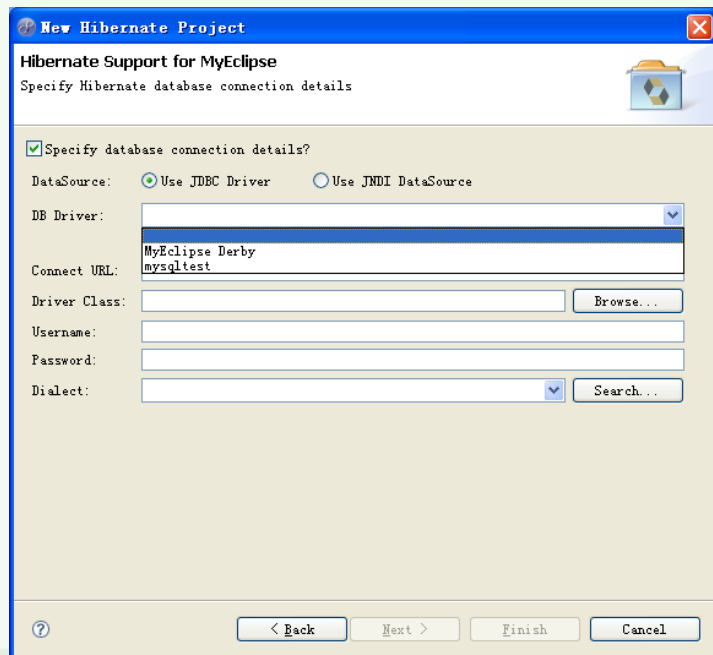


Hibernate 实例开发

三．为当前项目添加对 Hibernate 框架的支持

5. 点击【 Next 】，进入数据库连接信息配置的向导页

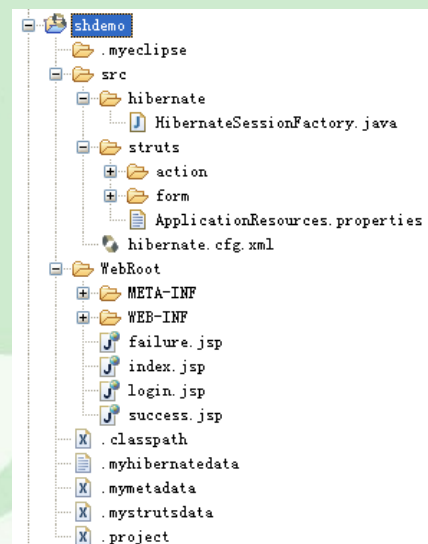
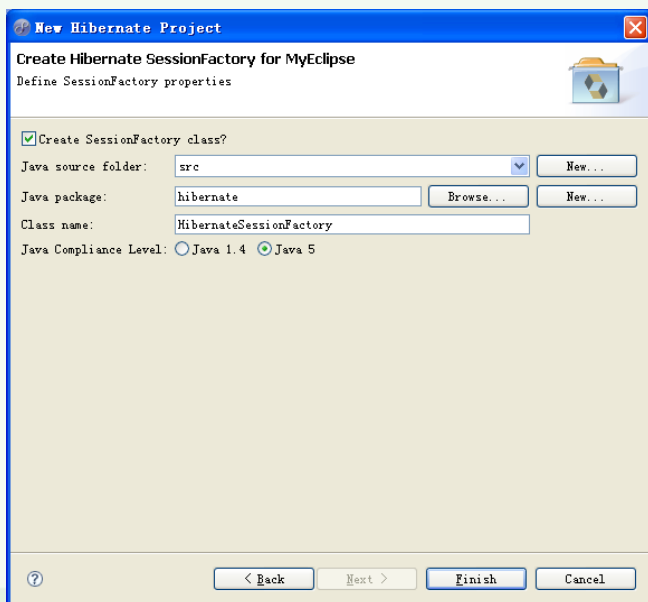
6. 在“ DB Driver” 下拉框中 “ mysqltest”



Hibernate 实例开发

三．为当前项目添加对 Hibernate 框架的支持

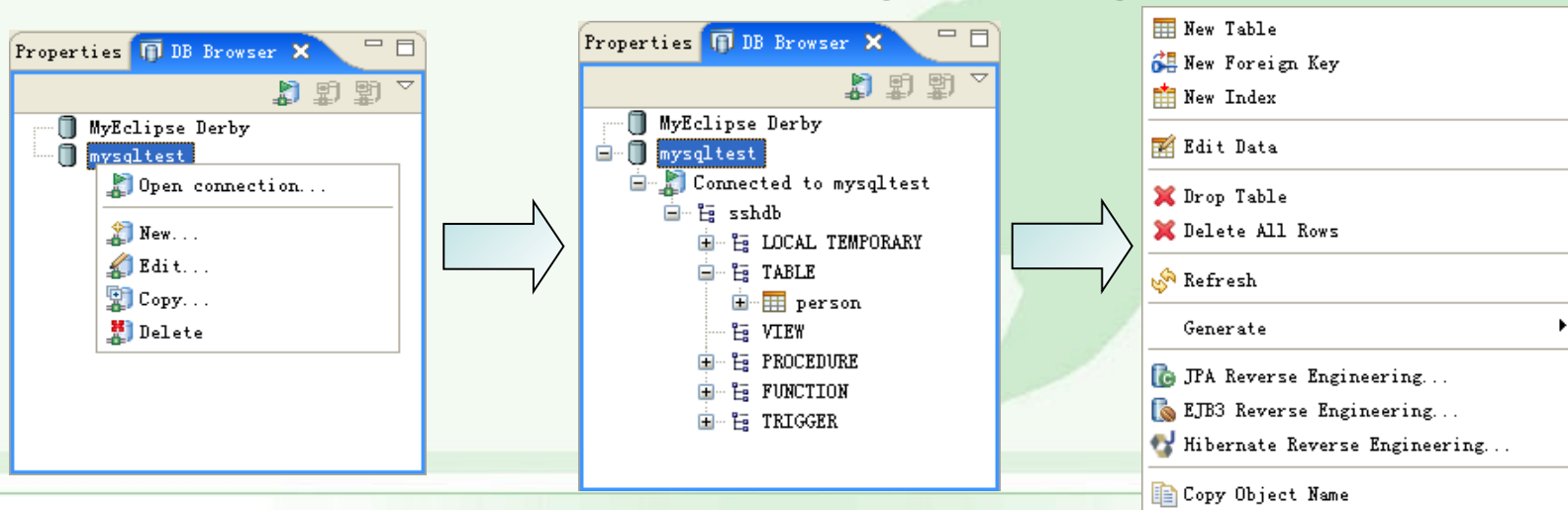
7. 点击【 Next 】，SessionFactory 必须建立在当前项目的“src”目录下的子目录中
8. 整个项目的物理文件结构如图：



Hibernate 实例开发

四 . 生成持久化类及 DAO

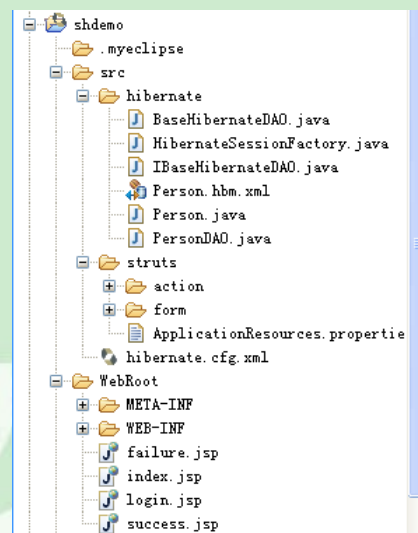
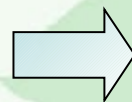
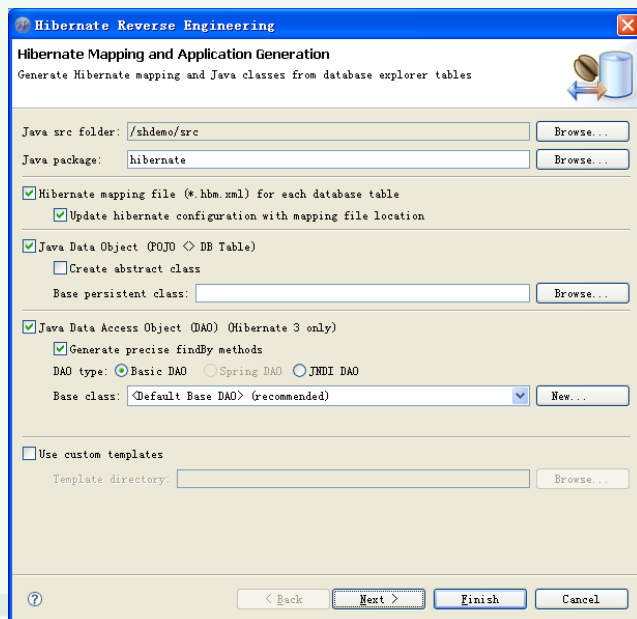
1. 选择“mysqltest”数据库，右击菜单中选择【 Open connection 】
2. 将数据库目录中的“TABLE”展开，直到显示出关系表
3. 选择“person”表，在右击菜单中进一步选择【 Hibernate Reverse Engineering 】



Hibernate 实例开发

四 . 生成持久化类及 DAO

4. 在“Java Package” 选择已经建立的名为“hibernate”的包，必须选中映射配置文件（ *.hbm.xml ）
5. 点击【 Finish 】，整个项目的物理文件结构如图：



Hibernate 实例开发

四 . 生成持久化类及 DAO

6. 在生成的文件中，在 PersonDAO.java 中，针对增删改的方法我们需要添加相应的事务处理代码，以确保所作的操作能立刻提交。以保存方法为例：

```
public void save(Person transientInstance) {  
    log.debug("saving Person instance");  
    try {  
        Transaction tc=getSession().beginTransaction();  
        getSession().save(transientInstance);  
        tc.commit();  
        log.debug("save successful");  
    } catch (RuntimeException re) {  
        log.error("save failed", re);  
        throw re;  
    }  
}
```


Hibernate 实例开发

五 . 在 LoginAction 中调用 Hibernate 的 DAO

1. 在 Struts 的 LoginAction 中调用 findByExample 方法

```
public List findByExample(Person instance) {  
    log.debug("finding Person instance by example") ;  
    try {  
        List results=getSession().createCriteria("hibernate.Person")  
.add(Example.create(instance))  
.list() ;  
  
        log.debug("find by example successful, result size: "  
                + results.size()) ;  
  
        return results ;  
    } catch (RuntimeException re) {  
        log.error("find by example failed", re) ;  
        throw re ;  
    }  
}
```

Hibernate 实例开发

五 . 在 LoginAction 中调用 Hibernate 的 DAO

2. 在先前已有的 LoginAction.java 中修改 execute 方法

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                             HttpServletRequest request, HttpServletResponse
response) {
    LoginForm loginForm = (LoginForm)form ;

    Person ps=new Person() ;
    ps.setUsername(loginForm.getUsername()) ;
    ps.setPassword(loginForm.getPassword()) ;

    PersonDAO dao=new PersonDAO() ;
    if (dao.findByExample(ps).size()>0) {
        return mapping.findForward("success") ;
    }
    return mapping.findForward("failure") ;
}
```

Spring 框架

Spring 框 架

- **Spring** 框架对 J2EE 思想进一步改造和扩充，使其发展成更开放、清晰、全面及高效的开发框架。一经推出，就得到众多开发者的拥戴。Spring 作为开源的中间件，贯穿视图层和持久层。然而，Spring 并不是要取代已有的框架，而是与这些框架进行无缝地整合。因此，Spring 框架所带的 JAR 包中，包含了很多对其它框架支持的类文件。

Spring 框 架

- 将 Spring 与我们刚学习的 Struts、Hibernate 框架相对比，他们的主要区别在于：
 - Spring 还可以集成其它框架，它是一个“大”框架
 - Struts 仅仅专注于实现 Web 应用程序的视图和控制器部分
 - Hibernate 仅仅专注于实现 Web 应用程序的数据持久层（模型层）

Spring 的依赖注入

- 依赖注入（ Dependency Injection ）是 Spring 的核心机制。当某个 Java 实例中需要调用其它 Java 实例时，创建被调用者的工作不需要由调用者来完成，而是由 Spring 容器来完成，然后将被调用者的实例作为调用者的一个 Javabeen 属性注入给调用者。
- **bean 是 Spring 管理的基本单位，任何 Java 对象都可被当成 bean 处理。**
- bean 的定义通常使用 XML 配置文件，正确定义的 bean 被 Spring 进行实例化以及依赖关系的注入。

Spring 的依赖注入

- bean 的定义通常使用 XML 配置文件。以下代码来自一个 Spring 配置文件 bean.xml，其中对 bean 实例，该 bean 实例是数据源，提供数据库连接。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans/spring-
beans-2.0.xsd">
  <bean id="DataSource"
        class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName"
              value="com.mysql.jdbc.Driver">
    </property>
    <property name="url"
              value="jdbc:mysql://127.0.0.1:3306/sshdb">
    </property>
    <property name="username" value="root"></property>
    <property name="password" value="test"></property>
  </bean>
```

```
</beans>
```

Spring 的依赖注入

- 通过以下 Java 程序由 ApplicationContext 来获取该 bean 的实例，获取实例时使用 bean 的唯一标识符：id 属性。该属性是 bean 实例在容器中的

```
public class BeanTest
public static void main(String[] args)throws Exception
// 实例化 Spring 容器，Spring 容器负责实例化 bean
ApplicationContext ctx =
new FileSystemXmlApplicationContext("bean.xml");
// 通过 bean id 获取 bean 实例，并强制类型转换为 DataSource
DataSource ds = (DataSource)ctx.getBean("dataSource");
// 通过 DataSource 来获取数据库连接
Connection conn = ds.getConnection();
// 通过数据库连接获取 Statement
Statement stmt = conn.createStatement();
// 使用 statement 执行 sql 语句
stmt.execute("insert into person (username,password) values('test','1234')");
// 清理资源，回收数据库连接资源
if(stmt!=null) stmt.close();
if(conn!=null) conn.close();
```


Spring 与 Hibernate 的整合

- 当 Hibernate 处于 Spring 的管理下，Hibernate 所需要的基础资源，都由 Spring 提供注入。
- Spring 提供了 DAO 支持，可以大大简化 DAO 组件的开发。
- Hibernate 的数据库访问需要在 Session 管理下，而 SessionFactory 是 Session 的工厂。SessionFactory 由 ApplicationContext 管理，并随着应用启动时自动加载。Spring 采用依赖注入为 DAO 对象注入 SessionFactory 的引用，这样 DAO 对象就可以对数据库进行访问操作。

Spring 与 Struts 的整合

- **ApplicationContext 对象是 Spring 的容器，负责管理所有的组件，从业务逻辑层组件到持久层组件，都必须运行在 Spring 容器中。spring 的配置文件 applicationContext.xml，其增加的配置信息**

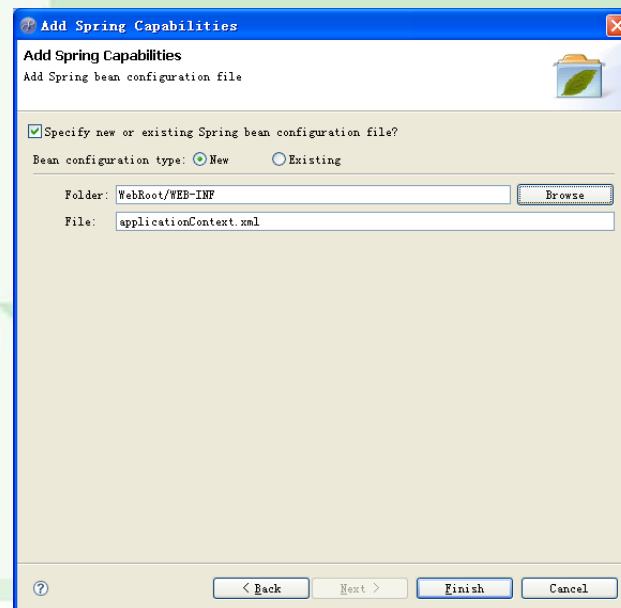
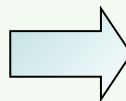
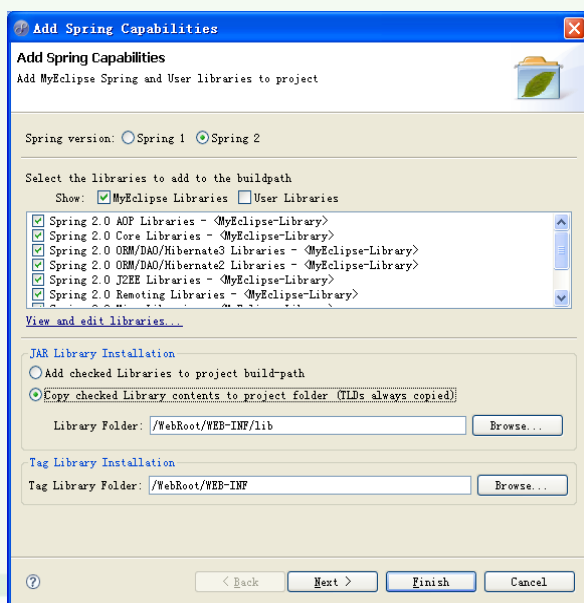
```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-
value>
</context-param>
<listener>
    <listener-class>
org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

- **监听器 ContextLoaderListener 根据 contextConfigLocation 的参数值来读取配置文件**

SSH 实例开发

一、为 Web 项目添加对 Spring 框架的支持

1. 选择 MyEclipse 的子菜单 Add Spring Capabilities，进入 Spring 配置向导页
2. 选择 Spring 的 JAR 包添加到当前项目的“/WebRoot/WEB-INF/Lib”下

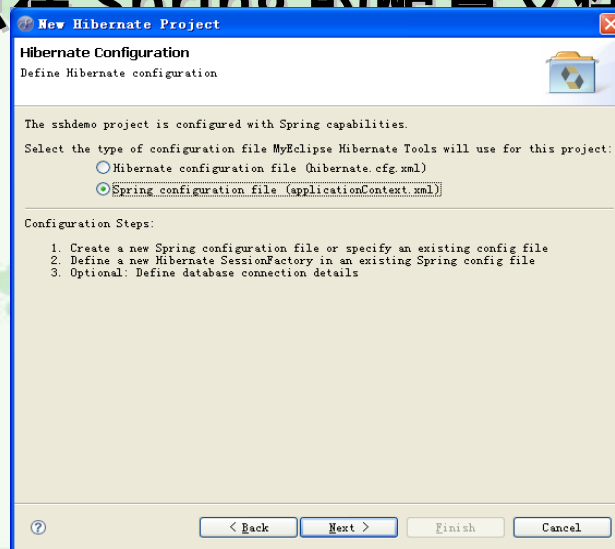
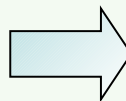
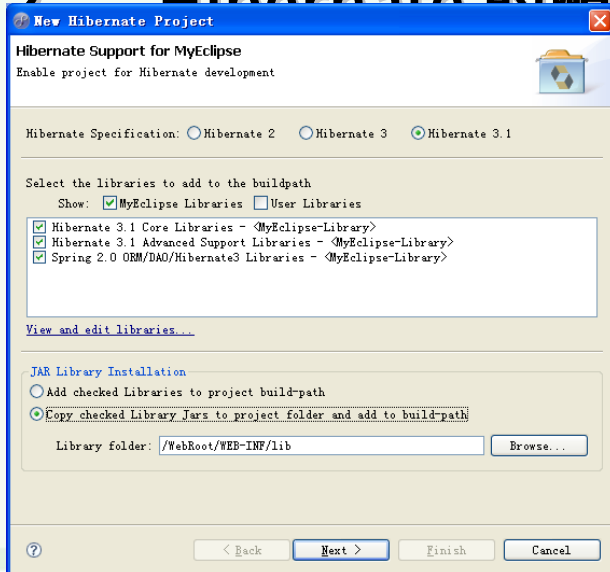


SSH 实例开发

二. 为 Web 项目添加对 hibernate 框架的支持

1. 在弹出的 Hibernate 配置向导对话框中, 将所有待选的 JAR 包添加到 “/WebRoot/WEB-INF/Lib” 下

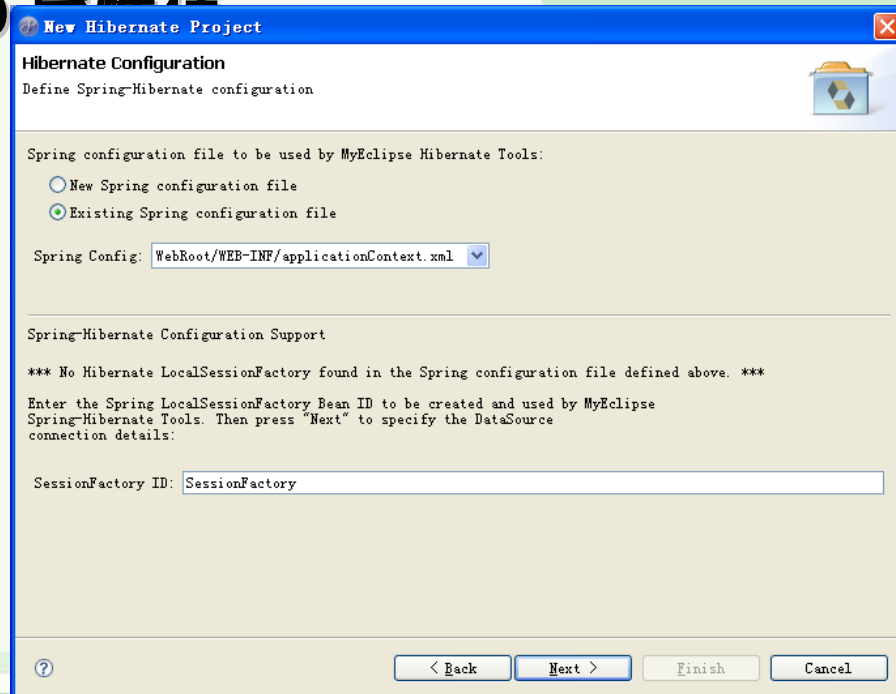
2. Hibernate 的配置信息在 Spring 的配置文件中



SSH 实例开发

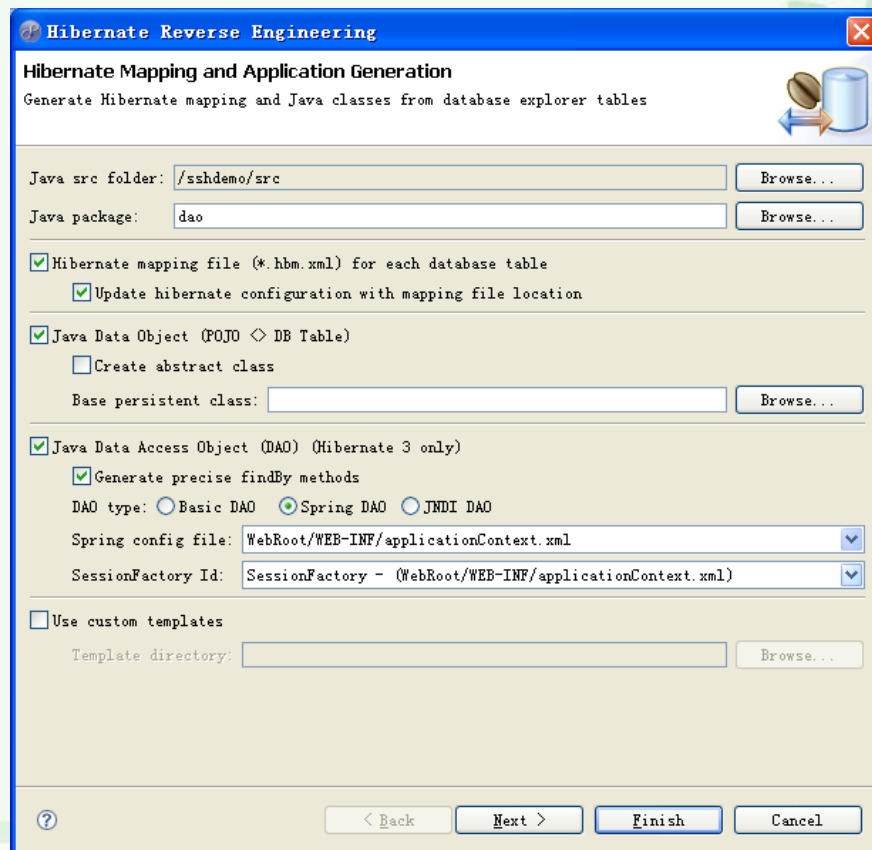
二. 为 Web 项目添加对 hibernate 框架的支持

3. 继续指定 applicationContext.xml 的相对路径，并为 SessionFactory 指定其在其中对应的 bean 的 ID。



SSH 实例开发

三．生成持久化类及 Spring DAO 类



SSH 实例开发

四．将 LoginAction 修改成 JavaBean 的格式

```
private Person person ;
private PersonDAO personDAO ;

    public Person getPerson() {
        return person ;
    }

    public void setPerson(Person person) {
        this.person = person ;
    }

    public PersonDAO getPersonDAO() {
        return personDAO ;
    }

    public void setPersonDAO(PersonDAO personDAO) {
        this.personDAO = personDAO ;
    }
}
```

SSH 实例开发

四．将 LoginAction 修改成 JavaBean 的格式

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                             HttpServletRequest request, HttpServletResponse
response) {
    LoginForm loginForm = (LoginForm) form ;

    person.setUsername(loginForm.getUsername()) ;
    person.setPassword(loginForm.getPassword()) ;

    if (personDAO.findByExample(person).size()>0) {
        return mapping.findForward("success") ;
    }
    return mapping.findForward("failure") ;
}
```


SSH 实例开发

五 . 修改 ApplicationContext.xml 配置文件

```
<bean id="DataSource"
      class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName"
      value="com.mysql.jdbc.Driver">
    </property>
    <property name="url"
      value="jdbc:mysql://127.0.0.1:3306/sshdb">
    </property>
    <property name="username" value="root"></property>
    <property name="password" value="test"></property>
  </bean>
  <bean id="PersonDAO" class="dao.PersonDAO">
    <property name="sessionFactory">
      <ref bean="SessionFactory"></ref>
    </property>
  </bean>
```

SSH 实例开发

五 . 修改 ApplicationContext.xml 配置文件

```
<bean id="SessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="dataSource">
        <ref bean="DataSource"></ref>
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.MySQLDialect
            </prop>
        </props>
    </property>
    <property name="mappingResources">
        <list>
            <value>dao/Person.hbm.xml</value>
        </list>
    </property>
</bean>
```

<bean id="Person" class="dao.Person" />

SSH 实例开发

五 . 修改 web.xml 配置文件

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-
value>
</context-param>
<listener>
    <listener-class>
org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

11.4 本章小结

- 本章依次对 **Struts**、**Hibernate** 以及 **Spring** 三个开发框架进行了介绍。在三个开发框架的具体介绍过程中，不仅对各框架自身的工作原理进行了明确的描述，而且在具体的实例开发过程中，重点探讨了如何用 MyEclipse 的特性实现框架之间的整合技术。希望读者通过对本章内容的学习，能够在今后的项目实践中选择相对成熟的第三方框架进行开发，这样会取得事半功倍的效果；同时要学会举一反三，对于当前以及未来出现的其它第三方开放框架能够快速上手。

本章习题

- 1、简述 Struts 框架的工作原理和开发步骤。
- 2、Struts 的视图层和控制器层分别包含哪些核心技术？
- 3、如何处理 Struts 的国际化问题？
- 4、简述 Hibernate 框架的工作原理和开发步骤。
- 5、在 Hibernate 中，持久化对象具有哪几种状态？如何在各种状态间进行转换？

本章习题

6. HQL 语句与 SQL 语句的编写方式上有什么差别？
7. Hibernate 中的条件查询涉及到哪些常用的接口和类？请分别举例说明。
8. 简述 Spring 中依赖注入的核心思想。
9. 简述 Struts、Spring、Hibernate 框架之间的联系。
10. 编写一个基于 SSH 框架的 Web 应用，实现对一个持久化对象完整的增删改和查询功能。